REMARKS

After the foregoing Amendment, claims 31-60 are currently pending in this application. Claims 1-30 have been canceled without prejudice. New claims 31-60 have been added to more distinctly claim subject matter which the Applicant regards as the invention. New claims 31-60 are similar in substance to claims 1-30 as originally filed. No new matter has been introduced into the application by these amendments.

Claim Objections

The Examiner objected to claim 17 because it is allegedly in improper form "because a multiple dependent claim [sic] 1 to 3 or 7 to 11," citing MPEP § 608.01(n). Claim 17 has been canceled; therefore the objection is moot. However, new claim 47 is in multiple dependent form. Applicant respectfully points out the new claim is in proper multiple dependent form, as was claim 17. MPEP § 608.01(n), cited by the Examiner, refers to 37 CFR 1.75(c), which states, "any dependent claim which refers to more than one other claim ("multiple dependent claim") shall refer to such other claims in the alternative only. A multiple dependent claim shall not serve as a basis for any other multiple dependent claim." (Emphasis added.) Thus, a multiple dependent claim is permissible, but a multiple dependent claim that depends from another multiple dependent claim is not. Claim 47 recites "a computer readable medium having computer readable instructions to instruct a computer to perform a method according to any one of claims 31 to 41." In accordance with 37 CFR 1.75(c), claim 47 is a multiple dependent claim that refers to more than one other claim in the alternative only, i.e., "any one of" claims 31 to 41. None of the claims depended from is itself a multiple dependent claim, and no other multiple dependent claim depends from claim 47. Therefore, claim 47 comports with 37 CFR 1.75(c), and it is in the form of a proper multiple dependent claim.

Claim 11 is objected to "because of the following informalities: undetermined a preamble portion of the claim." This objection is not understood. Nevertheless, claim 11 has been canceled, rendering the objection moot.

Claim Rejections - 35 USC § 101

Claims 12, 17-18, and 26 stand rejected under 35 USC § 101 as being allegedly directed to non-statutory matter. Claims 12, 17-18, and 26 have been canceled. Therefore, the rejection of those claims is moot.

Claim Rejections - 35 USC § 112

Claims 1, 12, 17, 18, and 26 stand rejected under 35 USC § 112 second paragraph as being allegedly indefinite. Claims 1, 12, 17, 18, and 26 have been canceled, so the rejection is moot. Nevertheless, the Examiner posed questions in paragraph 8 of the Action that suggest the Examiner may not fully appreciate the claimed invention. The Examiner is respectfully referred to the summary provided below in connection with the § 102 and § 103 rejections, as well as to the detailed description section of the application, for a fuller explanation of the claimed invention. In addition, the Examiner is kindly invited to contact at his convenience applicant's representative Michael Berman at 215-988-1164 for clarification, if needed.

Claim Rejections - 35 USC § 102, 103

Claims 18-19, 26-27 stand rejected under 35 USC § 102(e) as being allegedly anticipated by Stack (US Patent 6,257,774 B1, hereinafter "*Stack*"). Those claims have been canceled. In addition, claims 1-3, 7-13, 16-17, 20, 22-25, 28, and 30 stand rejected under 35 USC § 103(a) as being unpatentable over *Stack* (same as above) in view of Herrell (US 5,301,287), Kyker (US 6,467,027), Wesinger (6,052,788), or Kloth (6,598,034). Those claims have also been canceled,

mooting the rejections. Nevertheless, to advance prosecution of the application and examination of the new claims, applicant respectfully offers the following remarks.

The claimed invention is directed to data packet processing, such as would occur, for example, in a router on a network. Packet processing based on filtering according to a set of rules is widely known. However, problems can arise in connection with high-speed processing of packets according to complicated sets of rules. Updating a rule set can cause a pause in the operation of the packet processing engine, increasing latency and adversely affecting throughput of the engine, especially when the frequency of updates and/or the volume of processed packets is high.

The claimed invention can ameliorate such problems by managing compiled packet filtering rules in pieces, such as in code pages. Packet filtering rules can be compiled and made available to the packet processing engine as needed. Filter code can be updated by updating only a piece of the whole code. When a filter rule is changed, added, or removed, the claimed invention can change only the corresponding piece or pieces of code that are changed, added, or deleted. This allows for fast updates of the filter code. Shadow paging of packet filter code pages can provide consistency for the processing of packets whose processing has already begun, while parts of the filter code are being updated. As illustrated in particular in FIGs. 3a and 3b and explained on pages 12-13 with regard to an exemplary embodiment, when an update of the filter code occurs, the old filter code can still be kept available. Packets that have already begun to be processed at the time the update is initiated can continue to be processed using the old filter code, while newly received packets can be processed using the updated filter code. When the old code pages are no longer needed, they can be released from memory. In addition, frequent updating of the compiled rules can lead to a situation where more than two sets of code pages are in use at one time. Furthermore, the interval between subsequent filter rule updates can be shorter than the average processing time of a packet, whereby many updates can occur during the

average processing time of a packet. Multiple execution paths for multiple sets of packet filtering rules can be managed, as set forth in the claims.

In stark contrast to the claimed invention, *Stack* discloses a system for generating a computer application program and associated documentation. *Stack* is concerned with "the ability to design, implement, maintain and document an application with broad and highly configurable functionality," and is also concerned with the "training and documentation [to] deal with the manifold options that such applications include." (*Stack*, column 1 lines 24-29.) *Stack* generates high-level code that can then be compiled into an executable program, and also produces associated documentation. "Another advantage of the present invention is that the syntax generator provides for the autonomous generation of compilable or interpretable code fully consistent with the definition of a predetermined programming language." (*Stack*, column 4 lines 15-18.) Thus, *Stack* does not manipulate already-compiled pieces of code, and has nothing at all to do with packet filtering according to sets of rules that can be frequently updated, as do the claims. Furthermore, there is no way that *Stack* can be used to filter data packets in such an environment, or to manage compiled filter code in pieces for use in processing data packets.

The Examiner's reading of *Stack* is flawed. For example, the Examiner contends that *Stack* teaches processing packets according to a set of code pages at column 3 lines 55-65. That is incorrect. Instead, at the cited location *Stack* discloses a so-called rule processor which basically "unifies" so-called functionally descriptive atomic sequences (i.e., detailed descriptions of program functionality), in accordance with the known structure of a predetermined (i.e., selected) application program. Thereafter, a syntax processor produces "a coded representation of the structure and operation of the predetermined application program" in a predetermined (i.e., selected) programming language. (Column 3 line 64 – column 4 line 5.) Such programming languages include Basic, C, and C++. (Column 5 lines 20-24.) Thus, *Stack* discloses generating high-level computer code that can be compiled into an executable application program.

The Examiner also misreads *Stack* as disclosing creating a second set of code pages, citing column 5 line 43 – column 6 line 16. Instead, at the cited location *Stack* discloses producing modules of high-level (i.e., "compilable or interpretable") code and corresponding documentation. The modules can be assembled according to provided structures, and compiled into an application program, in a preferred embodiment into an accounting program. *Stack* discloses that an accounting program would include features such as transactional and detailed analysis screens (i.e., user displays), and reports including columnar reports, inquiry reports, and form reports, such as might, presumably, be used by an accountant.

Furthermore, the Examiner contended that *Stack* teaches a method for managing compiled filter code for processing data packets (which, as discussed above, *Stack* does not teach), wherein compiled code is managed in a plurality of pieces, again citing column 5 line 43 – column 6 line 16, as well as column 10 lines 27-61. (Applicant did not argue against the cited prior art at that time, relying instead on Examiner's representation that the claims contained allowable subject matter.) At those locations, the Examiner contended that *Stack* discloses "the method of divided [sic] the program rules or code into many sub-programs, segments, files and records." That may or may not be true, but it is not pertinent to claim 31 (or to claim 1 as originally filed), which recites that compiled code is managed in a plurality of pieces. *Stack* does not disclose managing compiled code in a plurality of pieces. *Stack* column 5 line 43 – column 6 line 16 discloses producing modules of high-level code and corresponding documentation, as discussed above. *Stack* column 10 lines 27-61 discloses the process whereby "the application structure and application sequences for a program have been constructed [and] an application author is invoked to generate a corresponding set of compilable or interpretable code." (Column 10 lines 27-31.) In other words, *Stack* therein discloses generating the high-level code from which an application executable can be compiled, presumably in a single operation, to produce an executable program. *Stack* does not disclose or suggest, at the cited locations or elsewhere, that the compiled code is managed in pieces, as does claim 31 (and the other independent

claims). Furthermore, none of the originally cited references supplement *Stack* to provide all of the features of claim 31 (or of the other independent claims).

It is well settled that a single reference or combination of references must teach every element or aspect of a claim in order to anticipate it or render it obvious under 35 USC § 102 or § 103, respectively. Because *Stack* does not disclose or suggest most or all of the elements contained in any of the claims, *Stack* cannot anticipate the claims.

In addition, none of the other cited references supplement *Stack* to provide all of the elements of any of the claims. Because the cited references, when read alone or in any possible combination, do not encompass within their scope nor teach all of the elements of any of the claims, a 35 USC § 103(a) rejection of any of the claims in view of the cited references cannot be supported.

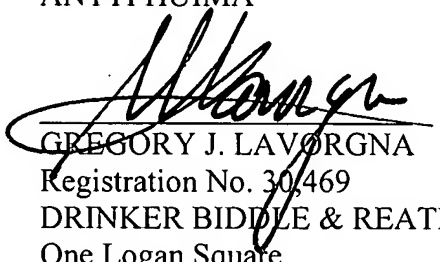
Conclusion

In view of the foregoing amendment and remarks, applicant respectfully submits that the present application, including claims 31 - 60, is in condition for allowance and an early notice of allowance is respectfully requested.

If the Examiner believes that any additional minor formal matters need to be addressed in order to place this application in condition for allowance, or that a telephone interview will help to materially advance the prosecution of this application, the Examiner is invited to contact the undersigned by telephone at the Examiner's convenience.

Respectfully submitted,

ANTTI HUIMA

BY: 

GREGORY J. LAVORGNA
Registration No. 30,469
DRINKER BIDDLE & REATH LLP
One Logan Square
18th and Cherry Streets
Philadelphia, PA 19103-6996
Tel: (215) 9880-3309
Fax: (215) 988-2757
Attorney for Applicant